# Project_Course8

*Jie Xue*

*June 1, 2016*

Clear the space

```
rm(list=ls())
cat("\014")
```

# Part 1. Data

First, loading the training data/testing data with replacing all missing with “NA”

```
Train<-read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!",""))
Test<-read.csv("pml-testing.csv",na.strings=c("NA","#DIV/0!",""))
```

Then, explore the data a little bit.

```
dim(Train)
```

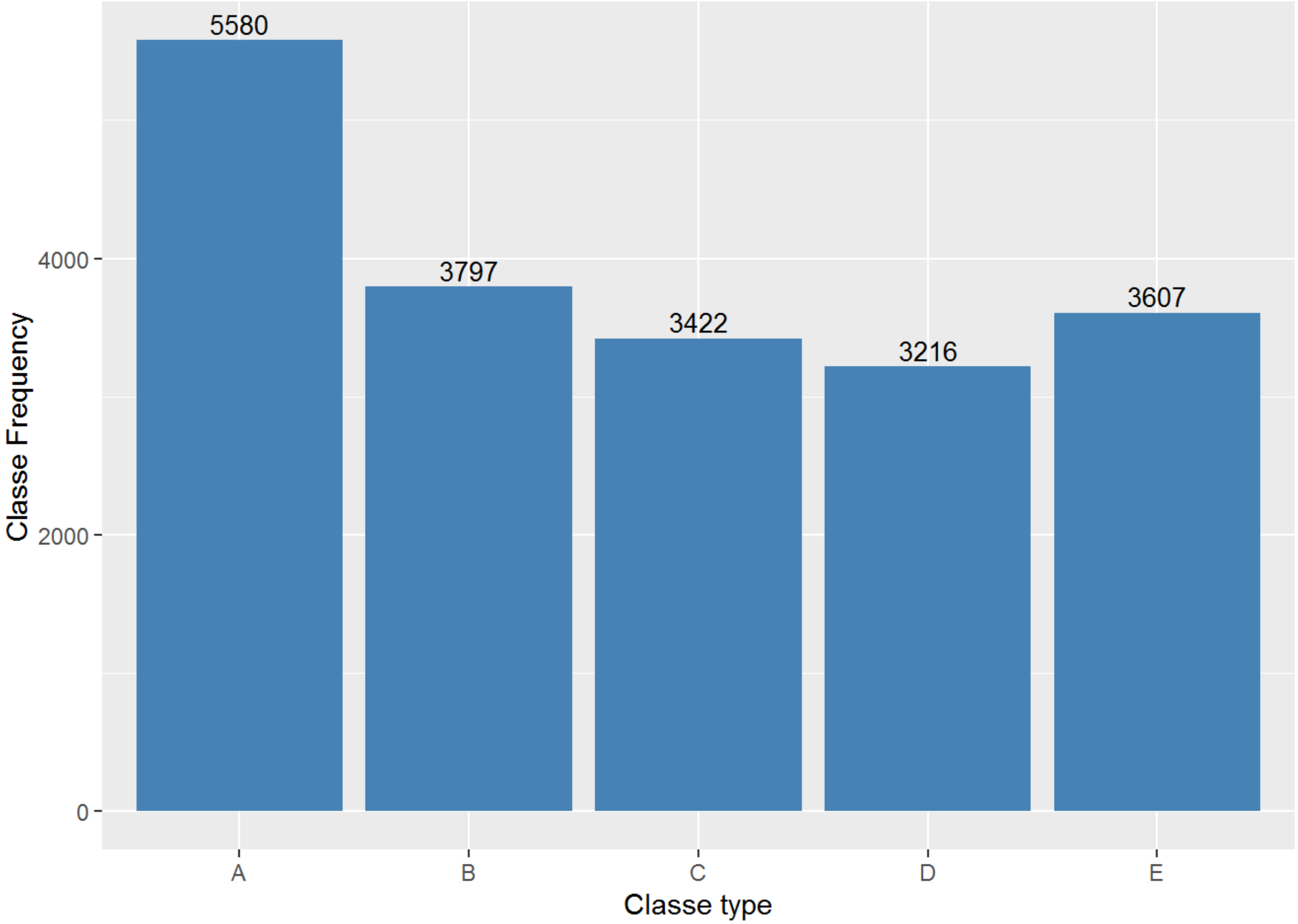
```
## [1] 19622   160
```

See, how classe distributed

```
Train.Class<-Train$classe
classe.freq<-table(Train.Class)
classe.freq<-as.data.frame(classe.freq)
```

Bar plot of classe with ggplot

```r
library("ggplot2")
p<-ggplot(classe.freq,aes(x=Train.Class,y=Freq),fill=Train.Class)+
    geom_bar(stat="identity", fill="steelblue") +
    geom_text(aes(label=Freq), vjust=-0.3,size=3.5)+
    ylab("Classe Frequency") +
    xlab("Classe type")
print(p)
```



# Part 2. Pre-Processing

Remove columns woth more than 50% NAs

```
Train <- Train[, colSums(is.na(Train)) < nrow(Train) * 0.5]
Test <- Test[, colSums(is.na(Test)) < nrow(Test) * 0.5]
```

Remove all Near Zero Variance variables

```
library(lattice)
library(caret)
NZV <- nearZeroVar(Train, saveMetrics= TRUE)
Train <- Train[,!NZV$nzv]
Test <- Test[,!NZV$nzv]
```

Remove unnecessary columns 1 to 6

```
Train<-Train[,-c(1:6) ]
Test<-Test[,-c(1:6) ]
```

Partition data into 60% and 40%

```
set.seed(123)
DTrain<-createDataPartition(Train$classe, p=0.7, list=FALSE)
Train.T<-Train[DTrain,]
Train.CV<-Train[-DTrain,]
```

# Part 3. Build prediction model

First, Try decision tree

```
Model.DT<-train(classe ~ ., method="rpart",data=Train.T)
```

```
## Loading required package: rpart
```

```
Prediction.DT <- predict(Model.DT, Train.CV)
```

Test results on our subTesting data set:

```
confusionMatrix(Prediction.DT, Train.CV$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  235   27   64   13
##          B  163  631   42  133  281
##          C  341  230  819  509  247
##          D  102   43  138  258   60
##          E    7    0    0    0  481
## 
## Overall Statistics
## 
##                Accuracy : 0.5523
##                  95% CI : (0.5394, 0.565)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4373
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6338   0.5540   0.7982  0.26763  0.44455
## Specificity            0.9195   0.8696   0.7269  0.93030  0.99854
## Pos Pred Value         0.7579   0.5048   0.3816  0.42928  0.98566
## Neg Pred Value         0.8633   0.8904   0.9446  0.86639  0.88864
## Prevalence             0.2845   0.1935   0.1743  0.16381  0.18386
## Detection Rate         0.1803   0.1072   0.1392  0.04384  0.08173
## Detection Prevalence   0.2379   0.2124   0.3647  0.10212  0.08292
## Balanced Accuracy      0.7767   0.7118   0.7626  0.59897  0.72154
```

The results are not good enough. Try another algorithm.

Second, try random forest

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
Model.RF <- randomForest(classe~.,data=Train.T)
Prediction.RF <- predict(Model.RF, Train.CV)
confusionMatrix(Prediction.RF, Train.CV$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    1 1133   11    0    0
##          C    0    0 1015   13    0
##          D    0    0    0  950    0
##          E    0    0    0    1 1082
## 
## Overall Statistics
## 
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9947   0.9893   0.9855   1.0000
## Specificity            0.9986   0.9975   0.9973   1.0000   0.9998
## Pos Pred Value         0.9964   0.9895   0.9874   1.0000   0.9991
## Neg Pred Value         0.9998   0.9987   0.9977   0.9972   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1925   0.1725   0.1614   0.1839
## Detection Prevalence   0.2853   0.1946   0.1747   0.1614   0.1840
## Balanced Accuracy      0.9990   0.9961   0.9933   0.9927   0.9999
```

Check the Importance with Overall>200

```r
importance <- varImp(Model.RF)
RN<-rownames(importance)
importance<-cbind(RN,importance)
library(dplyr)
```

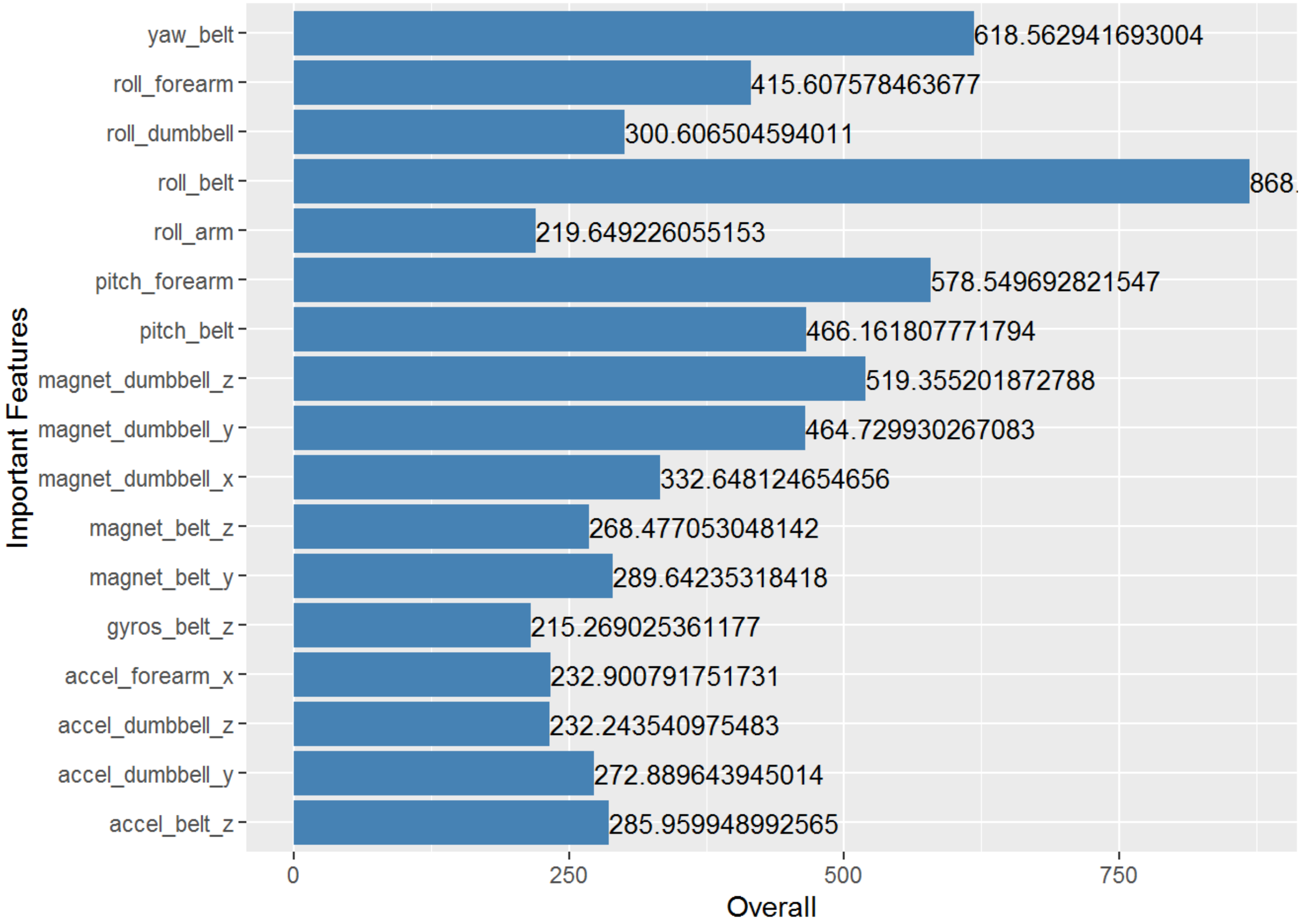
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
importance<-arrange(importance,desc(Overall))
importance<-filter(importance,Overall>200)
p<-ggplot(importance,aes(x=RN,y=Overall),fill=Train.Class)+
    geom_bar(stat="identity", fill="steelblue") +
    geom_text(aes(label=Overall),hjust=0,size=3.5)+
    coord_flip()+
    ylab("Overall") +
    xlab("Important Features")
print(p)
```



# part 4. Using the test data

```
Prediction.Test <- predict(Model.RF, Test)
Prediction.Test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```